Remarks

Claim 1 has been amended to move the qualifying phrase "if there was no data to be read from said pipe" from the end of the second step to the beginning of that step to make it clear that the condition refers to the result of the first step and that it qualifies both the issuing and terminating substeps and not just the terminating substep. Corresponding apparatus claim 8 and program product claim 11 have been similarly amended.

Claims 1 and 11 have also been amended to change "terminating" to "terminate" for proper parallel form ("having said first reader process issue . . . and then terminate")..

Both of these amendments are for purposes of clarification. No substantive change is intended.

Claims 1-6, 8-13 and 15-19 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over applicants' "admitted prior art" ("APA") in view of Stevens, UNIX Network Programming, pages 102-105 ("Stevens") (paper no. 5, page 1). Claims 7 and 14 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over this combination of references and in view of U.S. Patent 5,446,894 to DeMar et al. ("DeMar") (paper no. 5, page 2). These rejections are respectfully traversed.

Specific groups of claims are addressed below.

Claims 1-6 and 8-13

This group of claims contains three independent claims: 1, 8 and 11. It is therefore sufficient for the purposes of this action to address the rejection of these three claims on the combination of applicants' "admitted prior art" and Stevens.

Claim 1 is directed to a method for enabling the reading of data from a named pipe by a reader process while minimizing the use of system resources in an information handling system in which processes write data to and read data from a named pipe (Figs. 2A-2F: 204) by issuing function calls to an operating system (208). In accordance with the invention, a first reader process (202) issues a read function call to the operating system specifying the named pipe to attempt to read data from the pipe (Fig. 4: step 404). If there was no data to be read from the pipe (step 406), the first reader process issues an activate-on-receipt function call to the operating system specifying a new reader process (206) to be activated upon the receipt of data by the named pipe (step 408), then terminates (step 410). Claim 8 is similar to claim 1 but directed to apparatus, while claim 11 is similar to claim 1 but directed to a program storage device.

By allowing a reader to exit, applicants' invention as defined in these claims allows for the release of resources associated with the first process. At the same time, since the new reader process remains unactivated until data is received in the named pipe, the latter process does not use system resources associated with activation until they are needed. The present invention thus achieves its objective of minimizing the use of system resources.

As the Examiner notes, the background portion of applicants' specification describes a conventional information handling system in which processes write data to and read data from a named pipe by issuing write() and read() function calls to an operating system (page 1, lines 17-21). The Examiner correctly equates this description with applicants' preamble recitation of such an information handling system. The Examiner goes astray, however, in applying the other cited reference, Stevens.

Chapter 3 of Stevens deals generally with interprocess communication (IPC), and Section 3.4 (pages 102-109) deals with pipes as a particular mechanism for IPC. On page 102, Stevens describes a simple scenario of a pipe within a single process, with the same process both writing to and reading from the pipe. On pages 102-104, Stevens describes a more complex scenario in which a parent process creates a pipe (Fig. 3.4), then creates a copy of itself as a child process (Fig. 3.5), then closes its read connection to the pipe while the child process closes its write connection to create a one-way flow of data from the parent to the child (Fig. 3.6).

The Examiner equates the simple scenario on page 102 with applicants' claimed first step of having a first reader process issue a read function call to the operating system specifying a named pipe to attempt to read data from the pipe. Applicants do not dispute that this step is well known in the art.

The Examiner goes on, however, to equate the more complex scenario on pages 102-104 with the remaining part of applicants' preamble recitation ("a method of enabling the reading of data from a named pipe by a reader process while minimizing the use of system resources"), as well as with applicants' claimed second step of having the first reader process issue an activate-on-receipt function call specifying a new reader process to be activated upon the receipt of data by the named pipe and then terminate if there was no data to be read from the pipe. The Examiner concludes that it would have been obvious "to apply the teaching of Stevens to the system of APA", since one "would have been motivated to make such a modification in order to provide communication between processes" (paper no. 5, page 2). Applicants respectfully disagree.

Contrary to the Examiner's assertion, Stevens does not teach having a reader (1) issue an activate-on-receipt function call specifying a new reader process to be activated upon the receipt of data by a named pipe and then (2) terminate if there was no data to be read from the pipe. In the first place, in applicants' claimed system these substeps are contingent on a first reader attempting to read data from a pipe and there being no data to be read from the pipe. There is nothing of this contingent nature in the referenced portion of Stevens. Rather, as shown in Figs. 3.4 and 3.5 and the accompanying text, the parent process creates a child process and closes its write end of the pipe, regardless of the result of any attempted read operation.

Secondly, the operations performed by Stevens are completely different from the operations performed by applicants. In applicants' claimed system, if an attempted read operation is unsuccessful because there is no data to be read, the first reader process issues an activate-on-receipt function call to the operating system, specifying a new reader process to be activated upon the receipt of data by the named pipe, and then terminates. In the Stevens scenario, by contrast, the child process is created unconditionally, regardless of any pipe activity, new or old, and the parent process continues to run (so that there is an active process at both the write end and the read end of the pipe). Accordingly, not only does Stevens fail to perform the second step of applicants' claimed invention, but he also fails to achieve applicants' objective of minimizing resource usage.

Therefore, Stevens fails to teach the subject matter of applicants' invention as defined in claims 1, 8 and 11, either singly or in combination with applicants' "admitted prior art". Accordingly, claims 1-6 and 8-13, which are based upon these three claims, distinguish patentably over this combination of references.

Claims 7 and 14

Claim 7 is dependent on claim 1, but additionally recites the step of having the operating system activate the reader process in response to the activate-on-receipt function call upon the receipt of data by the named pipe. Claim 14 adds a similar limitation to program product claim 11.

DeMar discloses a method for controlling a multitasking time slice to support a desired system frame operating time. The patentee states that a process puts itself to sleep if no data is found in a pipe being read (col. 65, lines 47-51), to be reawakened when data later appears. This corresponds essentially to the blocking mode of operation described at page 1, lines 24-28, of applicants' specification, and does not represent new art in this respect. While DeMar may awaken an <u>existing</u> process upon the receipt of data, they do not activate a <u>new</u> process upon the receipt of such data as claimed by aplicants.

Claims 15-19

These claims are similar in tenor to claims 1-14, but are from the standpoint of the operating system. Thus, claim 15 is directed to a method in which an operating system receives an activate-on receipt function call from a first reader process specifying a new reader process to be activated upon the receipt of data by a named pipe, and then activates the new reader process in response to the activate-on-receipt function call upon the receipt of data by the named pipe. Claim 18 is similar but is directed to apparatus, while claim 19 is similar but directed to a program storage device.

Claims 15-19 distinguish over the references cited in a manner similar to that of claims 1-14. Thus, Stevens does not teach receiving an activate-on-receipt function call from a reader process, nor does he teach activating a new reader process in response to such call upon the receipt of data by a named pipe. Rather, at the conclusion of the scenario, both the parent process and the child process are active, regardless of the presence of any data in the pipe.

Conclusion

For the foregoing reasons, all of claims 1-19 as amended are believed to distinguish over the art cited by the Examiner. Applicants therefore respectfully request that the Examiner reconsider the application as amended and, upon such consideration, hold all claims allowable and pass the case to issue at an early date. Such action is earnestly solicited.

Respectfully submitted,
MIGUEL A. DELATORRE et al.

By [illegible]
William A. Kinnaman, Jr.
Registration No. 27,650
Phone: (845) 433-1175
Fax: (845) 432-9601

WAK/wak